# Towards Fast and Realistic Image Synthesis from Real Views

J. Blanc, R. Mohr
Project Movi, Gravir-Imag
Inria Rhône-Alpes
655, av. de l'Europe
38330 Montbonnot St-Martin
France
*Jerome.Blanc@inrialpes.fr*

**Abstract**

This paper is about scene modeling and image synthesis from existing views. From some known views of a 3d scene, called "reference views", we synthesize other views of the same scene, under another point of view. The outline of the whole process is simple: we first match the reference views, then we reconstruct the matched points in projective or Euclidean space. We build the new view by projecting these points onto the plane of a virtual camera. We don't need to calibrate the cameras, as we don't need a real 3d model of the scene to synthesize the new view (a projective model is enough; transfer can be achieved by using the trilinear tensor or plain projective re-projection).

This method is equivalent to a fast and realistic image synthesis. Fast, because we don't need to build a complex model of the scene by hand. Fast and realistic, because we don't have to render the synthesized view: the intensities or colors of the pixels are directly extracted from the (real) reference views. This technique has numerous potential uses in video compression, image synthesis, and more generally in all Virtual Reality applications.

**Keywords:** *point matching, automatic modeling, image synthesis from existing views.*

# 1 Introduction

## 1.1 Motivation

From some known views of a 3d scene, called "reference views", we synthesize other views of the same scene, under another point of view. That is, if we take pictures or video images of a scene, we can build new views, and for instance simulate a displacement inside the scene. This has numerous applications, among which:

- very high rate video compression: only the reference views and the displacement of the camera need to be transmitted. On the other end, the receiver can reconstruct all the other views. The amount of transmitted data doesn't depend on the size of the images.
- fast scene modeling: it is a tedious or even impossible task to model a complex real scene, containing trees, rocks... With this method, you just need to take pictures, and those pictures *are* the model.
- fast synthesis: the intensities and colors of the points of the synthesized views are directly extracted from the reference views, so they're as realistic as possible (they're real!) and we don't need to render the synthesized views.
- all domains of Virtual Realities, especially simulated navigation (e.g. for simulation).

All these potential applications led to growing interest on synthesis from real views, mostly people from the computer vision community ([Sha 94, Wer 94, Koc 95, Lav 94, Kan 95, Deb 96, Mes 96, Lec 97, Ois 96], see below) and sometimes from people working on image synthesis [Lev 96].

### 1.2 Short Review

[Sha 94] uses the trilinear constraints existing between 3 views. Knowing 2 views of a scene, the coordinates of two matching points $p_1$ and $p_2$ in these views are linked to the coordinates of the matching point $p_3$ in a third view. I.e. we have $p_3 = f(\alpha, p_1, p_2)$, where $\alpha$ represents the relative positions and orientations of the three viewpoints, in a projective way. This is mathematically equivalent to a projective reconstruction from 2 images, followed by a projection onto the third image. This method is not simply extendable to more than two reference views, and the $\alpha$ parameters are an uneasy way of specifying translations and rotations.

[Wer 94] synthesizes new views by a kind of morphing between two reference views, which does not ensure that the synthesized views will be physically-valid, especially if they don't lie in-between the reference views.

[Koc 95] synthesizes new views of an object. He needs to put the object onto a turntable, and uses a special shape-from-contours algorithm to reconstruct a 3D model of the object. This is limited to objects which can be put on a turntable.

[Lav 94] uses a projective, then affine, then Euclidean reconstruction. He matches some sparse points on the reference views and computes a model consisting of a textured triangle mesh. Matching seems to be done by hand.

[Kan 95] built a special dome with 51 suspended cameras pointing at the center and 51 synchronized VCRs recording the scene. A dense matching is then performed on the dynamic scene, using all 51 views, and the reconstructed cloud of 3D points can be viewed under any angle. This needs a huge structure, and again can not be applied to bigger objects, e.g. a landscape. More, the cameras need to be calibrated then fixed, which is not necessary.

[Deb 96] synthesizes new views of buildings from photos. The scene is modeled in textured prismatic blocks. The matching task is mainly manual, and peculiarly adapted to architectural rendering.

[Mes 96, Lec 97] and [Ois 96] use sparse (resp. dense) matches between the reference views, then build a model in textured triangles automatically. They're the most close to our work, also theirs is more oriented towards video compression.

### 1.3 Algorithm Outline

As could be imagined, the algorithm has the following structure:

1. **match the reference views (registration).** We compute correspondences between points of the reference views; this is a necessary step before we can infer any depth information.

2. **compute a model of the scene.** The model will be Euclidean or projective whether the cameras are calibrated or not. We'll distinguish two kinds of model: a simple one consisting of a dense set of points (cloud), and a more complex one consisting of 3D textured triangles (not much complicated to build, but hard to use correctly).

3. **synthesize the new views.** The computed model is simply projected onto the image plane of the virtual camera. If this is a Euclidean model (calibrated cameras), we can use standard tools such as VRML.

The three points will be detailed in the three following sections.

## 2 Matching the Reference Views

The first step is to match the $N$ reference views together. Since we want to synthesize geometrically valid views of the scene, we need some kind of depth information (either projective or Euclidean); starting from images only, matching these images together is a unavoidable step to get this information. To do so, we use basic algorithms on two images, then we merge the binocular matches to get $N$-ocular matches.

We use a two pass algorithm: a first unconstrained matching step between trackable points allows us to compute the epipolar geometry between the two views, and a second epipolar-constrained registration gives us many more (and more reliable) matches.

### 2.1 Sparse or Dense Matches?

Most people e.g. [Kan 95] match as many points as possible between the reference views and compute a dense disparity map. This approach is suitable for synthesizing new views, but not for handling the computed model of

the scene. Another approach is to match some sparse points in the images, and to assume the other points are on flat surfaces. This gives us realistic synthetic images as well, under certain circumstances. We will come back to these two points later.

We present below a two stage matching leading to either dense or sparse matches.

## 2.2 Matching Two Views

### 2.2.1 Unconstrained Matching

We first extract interest points from the reference views. These points need to be recognizable enough for the algorithm to be able to track them in the views. That's why we use interest points (mainly corners) detected by a Harris 2D filter. This filter was slightly enhanced by [Sch 96] to improve repeatability: the detected point is not always right on a corner (1 or 2 pixels apart), but will always be on the same side of the "real" corner. It becomes more likely that we are detecting (and tracking) the projection of a real 3D point.

These points are then matched together using correlation measures. Using standard correlation measures makes no sense, because the detected Harris points frequently lie on occluding edges. That's why we use robust correlation measures as described in [Lan 96a]. The robust correlation is similar to a plain correlation measure like ZSSD, which is a centered sum of the squared differences of the pixels' values. But in the robust ZSSD, we only take into account the "not too different" pixels. In every correlation mask, the most dissimilar pixels are not included in the total sum. This way, occluded or occluding pixels are discarded, and this avoids the known problem of computing correlations on occluding contours.

This criterion is cross-checked in the two images: for one point in image 1, we seek the best match in image 2; for this last point we seek the best match in image 1, and if we find the first point again, then we keep the match, else we reject it.

As the Harris detector is not too precise (at best, 1 pixel), we need to refine the matches. This is done by simple recursive relocation, computing correlations on non-integer pixels[1] in an area of $1 \times 1$ sq-pixels and keeping the best.

The last step is to compute the epipolar geometry between the two images; we do this with a robust normalized algorithm combining the approaches of [Har 95] and [Zha 94]. This robust algorithm can cope with false matches, and its principle is the following: we perform many random trials, compute the epipolar geometry with the given points, then compute the median error relative to the other points. Among all these trials, we only keep the one leading to the minimal median error. The precision of the computed epipolar geometry is better than 1 pixel.

### 2.2.2 Epipolar-Constrained Matching

Using the previously computed epipolar geometry as a constraint in the matching process gives us many more matches, and more reliable, since the matching points are only searched for along their conjugated epipolar lines.

On the two reference views shown in figure 1, we use this constraint along with a Dynamic Programming matching algorithm (see for instance [Oht 85, Gei 92]). We get a dense disparity map shown in figure 2. It is to be noticed that no interline cohesiveness constraint was used in this process, and faults are visible on the edge of the cup. The points on the table were removed, because there was no texture to match these points.

As we will see in the following sections, sparse matches can be enough to synthesize new views. To compute sparse matches, we just keep points on edges, e.g. detected by a Heitger 1D filter along conjugated epipolar lines. We cannot use a Harris detector, as it detects only points with a strong local curvature, i.e. corners; at this stage we don't need only corners, but also points on edges. In fact we assume that an edge in an image corresponds to a 3D surface disparity change. All other points are assumed to lie on flat surfaces (planes) lying between the edges.

For the same reason as before, the precision of the matches is at best 1 pixel. As we are planning to reconstruct 3D points (either in projective or Euclidean space) from these matches, we have to refine them again to gain a subpixel accuracy. If we only have sparse matches, the same iterative search can be applied again. But if we have dense matches (about the number of points in the images), such a method would last for hours, and we use the non-iterative subpixel refinement described in [Lan 96b][2]. This methods makes the assumption that the intensity of the pixels in image 2 is locally a bilinear function of the intensities in image 1 and of the disparity $\delta$ between the two patches. By estimating this function from the pixel data in image 1 and image 2, we estimate the (sub-pixel) disparity $\delta$.

[1] The pixel value on non-integer pixels is computed by bilinear interpolation among the 4 nearest neighbours.

[2] A more precise method taking into account the local shape of the surface can be found in [Lan 96c].

# 3 Building a Model

## 3.1 From Dense Matches

From dense matches, we can easily compute the set of 3D points constituting the model of the observed scene. If we have no other information, knowing the epipolar geometry is enough to be able to compute the 3D points in a projective frame: from the fundamental matrix can be extracted two compatible projection matrices into image 1 and image 2; for every matched couple in the images, we draw the two viewing lines passing through the two centers of projection of the cameras; their intersection is a 3D point of the scene. From the dense set of matches, we get a dense set of 3D points.

This set is unviewable as is, but can be projected onto the image plane of a virtual camera to produce a "real" image. The problem is that it is uneasy to specify the position and orientation of the virtual camera in a projective frame, as angles and distances are not defined (we can still specify the point of view of the image to synthesize by setting the positions of the projections of six points in the image). A more easy way is to achieve complete calibration; what we do is entering the internal parameters of the camera into the system, which computes a complete calibration using the computed epipolar geometry as in [Luo 92]. Another approach is auto-calibration, but we need at least 3 images. We are currently implementing an algorithm working on 4 images imagined by [Tri 96].

The advantage of using a dense model is that any outlier (false match) will be lost in the crowd and become unnoticeable. That is, from almost any view point we'll get an acceptable synthesized view.

The drawback is that we only have a cloud of 3D points, with no topological relation between them. As we didn't capture all points, we will get holes in the synthesized views, and the pixels will part when we get nearer the scene, letting the pixels behind show through. A more important drawback is that this model will not be easy to handle and edit; as we lost all information e.g. on connexity, how could we remove a plane from the computed model?

## 3.2 From Sparse Matches

From sparse matches it is only possible to compute a sparse set of 3D points. Let's assume we are in a Euclidean frame. As we can not just project the set of points to get a synthesized view (we would only get the contours), and under the assumption that the unmatched points lie on planes, then what we need is a tessellation of the 3D points.

Of course, a 3D tessellation of the points is a real hard problem as we don't know where the 3D surfaces lie between the points. For instance, the four vertices of a tetrahedron can belong to 1 to 4 different surfaces. Although which surfaces to keep could be decided by examining what is really seen or occluded in the reference views, we believe this approach is too difficult, and we decided to compute a 2D tessellation.

What we do is simply computing a 2D Delaunay triangulation in one of the reference views. We choose as vertices of the triangulation the projections of known 3D points. The 2D triangulation we obtain is a 2D tessellation of the image. By assigning to each vertex the depth computed from the precedent step, the 2D tessellation of the image becomes a 3D tessellation of the scene. Such a triangulation can be seen in figure 3, along with the corresponding 3D model.

The advantages are speed and simplicity; but as can be seen on these figures, the correctness of the matches becomes of great importance. False matches are exacerbated, because they're now vertices of the triangulated model. Indeed the synthesized views will be valid only under a certain angle, and will become unacceptable if we move too far away from the reference view which was triangulated.

We are currently investigating a way to determine whether a view is acceptable or not. Once we have this criterion, we can use it to decide which model to use depending on where the virtual camera lies. That is, we compute several models from the tessellation of all the reference views (we have at least $N$ models), and knowing the position of the observer, we decide to switch to the "nearest" model, the one leading to the most acceptable synthesized view.

# 4 Synthesizing the New Views

## 4.1 From Dense Matches

The model is a 3D projective or Euclidean set of points. The intensities and colors of these points are mixed from the colors of the matches in the original views. We project this set of colored points onto the plane of the virtual

camera.

Some examples of synthesized views are shown on figure 4. The holes were filtered out by a plain $5 \times 5$ median filter. Still, big holes remain.

### 4.2 From Sparse Matches

The model in textured triangles is stored as a VRML file, which can be viewed with any standard viewer (we used VRWEB). As the triangles are filled with the texture extracted from the reference views, we don't have problems with holes or pixels parting; but false matches are exacerbated, as shown on figure 5 (edges of the starfruit), if we go too far away from the original viewpoint. As noticed in the previous section, a solution is to build many models, from all the original viewpoints, and to switch to the most adequate model. This requires to write an entirely new viewer, as standard viewers can not be used any longer.

## 5 Conclusion

We explored the ways to synthesize new views from existing views. The two main methods are projecting a dense cloud of 3D points onto a plane, or using a model made of textured triangles. The former is the most widely used technique, but requires a dense matching step, and leads to a model which can not be easily handled nor edited. The latter requires only sparse but precise matches, on some selected points (edges).

What remains to be done is:

- improve the matching, to get more precise and especially more reliable matches,
- define a criterion do decide whether a synthesized view is of acceptable quality or not (this is probably the hardest step, and we're working with people from image synthesis on a mathematical and perceptual basis),
- based on this criterion, develop a multi-model viewer, which displays the model adapted to the position where the observer stands.

Since we compute realistic views without manual modeling, the applications of this technique will be numerous in all the usual domains of image synthesis and Virtual Realities.

## References


[Deb 96] P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. Technical Report CSD-96-893, University of California, Berkeley, January 1996. also published in SIGGRAPH'96.

[Gei 92] D. Geiger, B. Ladendorf, and A. Yuille. Occlusions and binocular stereo. In G. Sandini, editor, *Proceedings of the 2nd European Conference on Computer Vision, Santa Margherita Ligure, Italy*, pages 425–433. Springer Verlag, 1992.

[Har 95] R. Hartley. In defence of the 8-point algorithm. In *Proceedings of the 5th International Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 1064–1070, June 1995.

[Kan 95] T. Kanade, P.J. Narayanan, and P.W. Rander. Virtualized reality: Concepts and early results. In *Workshop on Representation of Visual Scenes, Cambridge, Massachusetts, USA*, pages 69–76, June 1995.

[Koc 95] R. Koch. 3D surface reconstruction from stereoscopic image sequences. In *Proceedings of the 5th International Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 109–114, June 1995.

[Lan 96a] Z.D. Lan and R. Mohr. Appariement robuste par correlation partielle. In *Journées ORASIS 1996, Clermont-Ferrand, France*, pages 99–104, 1996.

[Lan 96b] Z.D. Lan and R. Mohr. Direct linear sub-pixel correlation by incorporating neighbour pixels' information: a robust and precise matching method, June 1996. Internal note (unpublished).

Figure 1: Our two reference views.

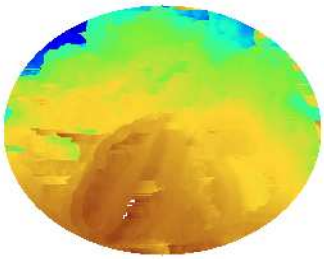

Figure 2: Dense disparity map (the warmer/darker, the nearer); white points were unmatched.

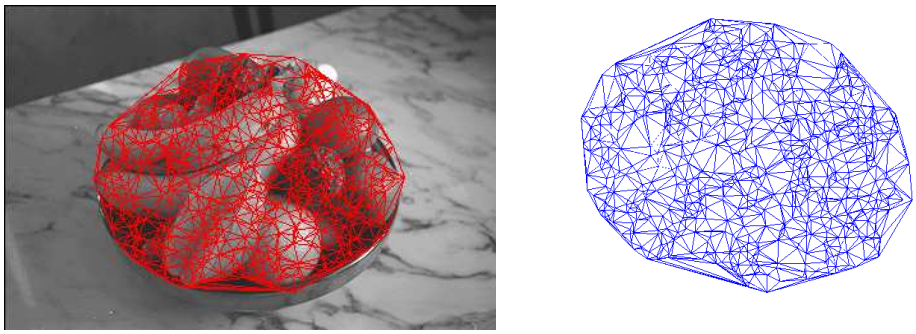

Figure 3: Delaunay triangulation in view #1, and the corresponding 3D model.

Figure 4: Some synthesized views using dense matches.

[Lan 96c] Z.D. Lan and R. Mohr. Precise matching by robust estimation of deformation and local coherence, 1996. Submitted to Cvpr97.

[Lav 94] S. Laveau and O. Faugeras. 3D scene representation as a collection of images and fundamental matrices. Technical report, INRIA, February 1994.

[Lec 97] P. Lechat, G. Le Mestre, and D. Pelé. An approach for scene reconstruction from the analysis of a triplet of still images. In *Electronic Imaging 1997*, 1997. Yet unpublished.

[Lev 96] M. Levoy and P. Hanrahan. Light field rendering. In SIGGRAPH *1996, New Orleans*, pages 31–42, 1996.

[Luo 92] Q.T. Luong. *Matrice fondamentale et autocalibration en vision par ordinateur.* Thèse de doctorat, Université de Paris-Sud, Orsay, France, December 1992.

[Mes 96] G. Le Mestre and D. Pelé. Trinocular image analysis for virtual frame reconstruction. In *VCIP 1996, Orlando*, 1996.

[Oht 85] Y. Ohta and T. Kanade. Stereo by intra and inter-scanline search using dynamic programming. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 7(2): 139–154, 1985.

[Ois 96] L. Oisel, L. Morin, B. Gasnier, and C. Labit. Application de la géométrie projective à la compression en TV3D. In *Journées ORASIS 1996, Clermont-Ferrand, France*, pages 129–134, May 1996.

[Sch 96] C. Schmid and R. Mohr. Image retrieval using local characterization. In *Proceedings of the* IEEE *International Conference on Image Processing*, volume II, pages 781–784, September 1996.

[Sha 94] A. Shashua. Trilinearity in visual recognition by alignment. In Jan-Olof Eklundh, editor, *Proceedings of the 3rd European Conference on Computer Vision, Stockholm, Sweden*, pages 479–484. Springer Verlag, May 1994.

[Tri 96] B. Triggs. Euclidean reconstruction, May 1996.

[Wer 94] Th. Werner. Rendering real-world objects without 3D model. Technical report, Czech Technical University, Dept. of Control, Faculty of Electrical Engineering, Czech Technical University, Karlovo nám. 13, 12135 Praha, Czech Republic, 1994.

[Zha 94] Z. Zhang, R. Deriche, O. Faugeras, and Q.T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. Rapport de recherche 2273, INRIA, May 1994.

Figure 5: Some synthesized views using sparse matches.